***CSC 3020***

***Java Programming***

**Winter Term 2021**

**Project 01**

**50 points**

**Due 03/23/2021 (11:45 A.M.)**

**The goal of this Project is:**

1. Being able to Analyze, Design, implement, and test a practical real-world application.
2. Being able to use selections and repetition structures
3. Being able to deal with methods
4. Being able to manipulate 1D & 2D arrays.
5. Learn about user-defined classes

**Requirements:**

* In a word file:
  + Analyze each problem; outline the problem and its solution requirements. (Describe the problem including input and output in your own words.))
  + Design an algorithm to solve the problem. (Describe the major steps for solving the problem.)
    - Use UML class diagram to model the class in question 2.
* Using java IDE software, implement the algorithm.
* Test the code for each problem and verify that the algorithm works; include a screenshot of each program output.

**Restrictions:**

You must work individually. Use only material from class or from the text book (chapters 1- 9). All code must be the work of the individual. Do not share your code or copy from external resources.

**Submission**

Submit 3 .java files (1 file for Q1 and 2 files for Q2); upload all files to the Canvas by the due date. DO NOT Email your files.

**Grading:**

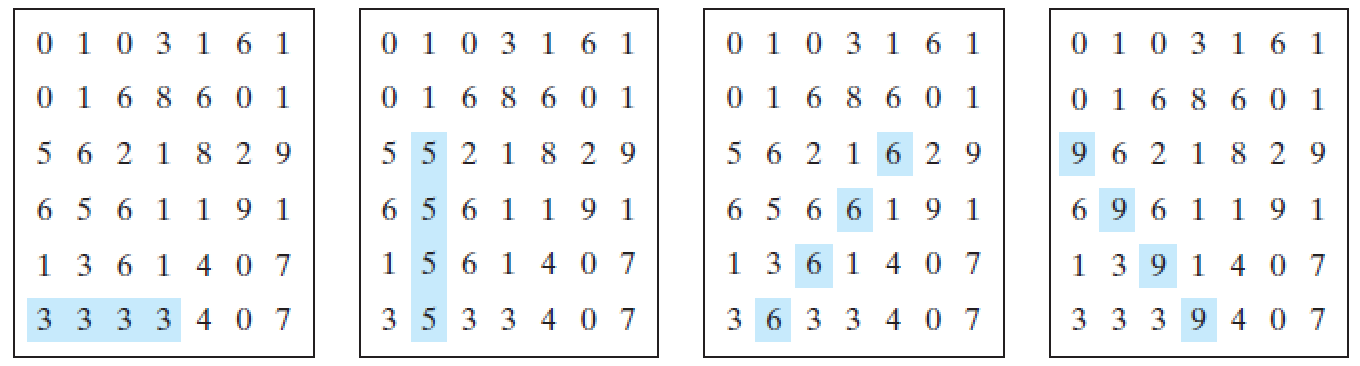
The grade of each program will be based on the creation of a program that works correctly, up to some details (50%), clear problem analysis and algorithm design (5%), the appropriate use of classes, methods and loops (25%), the production of clear output, with readable formatting and without unnecessary repetition (10%), composition of informative comments (5%), and testing the program with different inputs (5%). Programs must run.

**Question 01 - *Pattern recognition: four consecutive equal numbers* (25 points)**

Write the following method that tests whether a two-dimensional array has four consecutive numbers of the same value, either horizontally, vertically, or diagonally:

**public static boolean** isConsecutiveFour(**int**[][] values)

Write a test program that prompts the user to enter the number of rows and columns of a two-dimensional array then the values in the array, and displays true if the array contains four consecutive numbers with the same value. Otherwise, the program displays false. Here are some examples of the true cases:



***Question 02 - ATM machine* (25 points)**

Write a class named **Account** that contains:

* A private **int** data field named **id** for the account (default **0**).
* A private **double** data field named **balance** for the account (default **0**).
* A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0**). Assume that all accounts have the same interest rate.
* A private **Date** data field named **dateCreated** that stores the date when the account was created.
* A no-arg constructor that creates a default account.
* A constructor that creates an account with the specified id and initial balance.
* The accessor and mutator methods for **id**, **balance**, and **annualInterestRate**.
* The accessor method for **dateCreated**.
* A method named **getMonthlyInterestRate()** that returns the monthly interest rate.
* A method named **getMonthlyInterest()** that returns the monthly interest not the interest rate.
* A method named **withdraw** that withdraws a specified amount from the account.
* A method named **deposit** that deposits a specified amount to the account.

Note:

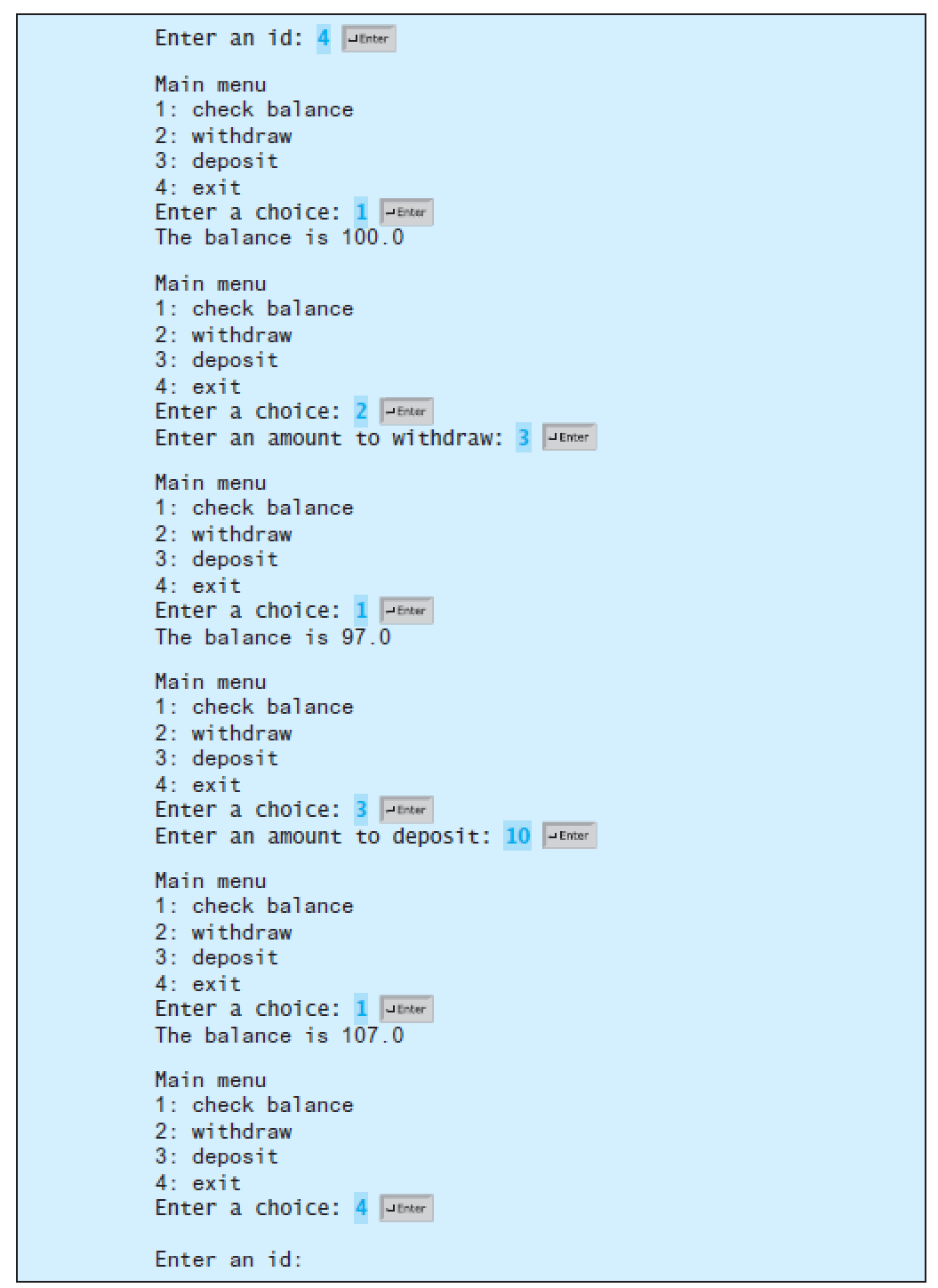
Monthly interest is **balance \* monthlyInterestRate**.

**monthlyInterestRate** is **annualInterestRate / 12**.

**annualInterestRate** is a percentage,for example 4.5%. You need to divide it by 100.

Draw the UML diagram for the class then implement the class.

Create another class (TestAccount) that usesthe **Account** class created above to simulate an ATM machine. Create 10 accounts in an array with id **0**, **1**, . . . , **9**, and an initial balance of $100. The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, the main menu is displayed as shown in the sample run. You can enter choice **1** for viewing the current balance, **2** for withdrawing money, **3** for depositing money, and **4** for exiting the main menu. Once you exit, the system will prompt for an id again. Thus, once the system starts, it will not stop.



|  |
| --- |
| **Account** |
| private id: int  private balance: double  private annualInterestRate: double; |
| Account()  Account(ID: int, bal: double)  setId(val: int) : void  getId(): int  setBalance(bal: double): void  getBalance(): double  setAnnualInterestRate(air: double): void  getAnnualInterestRate(): double  getMonthlyInterestRate(): double  getMonthlyInterest(): double  withdraw(): void  deposit(): void |